AN FPGA IMPLEMENTATION OF WALSH-HADAMARD TRANSFORMS FOR SIGNAL PROCESSING

*A.Amira, A.Bouridane, P.Milligan and M.Roula

School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN, United Kingdom
*A.Abbes@qub.ac.uk

ABSTRACT

This paper describes two approaches suitable for an FPGA implementation of Walsh-Hadamard transforms. These transforms are important in many signal-processing applications including speech compression, filtering and coding. Two novel architectures for the Fast Hadamard Transforms using both systolic architecture and distributed arithmetic techniques are presented. The first approach uses the Baugh-Wooley multiplication algorithm for a systolic architecture implementation. The second approach is based on both distributed arithmetic ROM and accumulator structure, and a sparse matrix factorisation technique. Implementations of the algorithms on a Xilinx FPGA board are described. Distributed arithmetic approach exhibits better performances when compared with the systolic architecture approach.

1. INTRODUCTION

Transform methods are useful in many types of applications, particularly if the features of interest can be characterised in the transform domain. A useful transform in speech and image processing is the Walsh-Hadamard transform (WHT)[10]. It is an orthogonal transform, with only additions and subtractions required, and is faster than sinusoidal-like transforms [3]. It can also be formulated as a matrix-vector multiplication similar to the Discrete Fourier transform DFT, and it has a fast algorithm which has $N\log_2 N$ additions and subtractions for N-point input samples [3], [7]. The Fast Walsh Hadamard Transform (FHT) is similar to the DFT and is globally recursive requiring global communication for the shuffling between different stages of the process.

As Field Programmable Gate Arrays (FPGAs) have grown in capacity, improved in performance, and decreased in cost, they have become a viable solution for performing computationally intensive tasks, with the ability to tackle applications for custom chips and programmable digital signal processing (DSP) devices [8].

It is the aim of this paper to develop efficient architectures, ideally suited for a fast computation of the FHT. Distributed Arithmetic (DA) and Systolic Architecture (SA) techniques have been described for the implementation of the FHT. Matrix factorisation methods and the symmetry of the Hadamard matrices have been exploited to develop the mathematical model in order to reduce the ROM size, the area consumed by the design, and to speed up the computation procedure by minimising the number of addition and subtraction operations required in the case of DA technique [5]. The Baugh-Wooley algorithm has been used for the implementation of the systolic architecture, this type of multipliers is characterised by the simplicity, regularity and modularity of the structure [1], [2].

The architectures proposed in this paper have been designed and targeted to the Xilinx XCV1000E of the Virtex-E family which has the following important features [8]:

- Fast, and high-density Field-Programmable Gate Array;
- Flexible architecture that balances speed and density; and
- Built-in clock- management circuitry.

The composition of the rest of the paper is as follows. The mathematical model for the 1-D FHT algorithm is given in section 2. Section 3 is concerned with the proposed architectures using both (SA) and (DA) techniques. The analysis of the implementation results obtained is given in section 4. Concluding remarks are given in section 5.

2. MATHEMATICAL MODEL

Typically, a Hadamard matrix is defined iteratively as:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}.$$

where H_N is a Hadamard matrix of size NxN. Furthermore, if the transform length N is a power of two (i.e., $N = 2^p$), then a (FHT) algorithm (similar to Fast Fourier Transform (FFT)) can be used for its fast computation.

2.1. 1-D FHT based Systolic Architecture (SA)

Let the input data and the transformed data be represented by the two vectors X and Y of size N, respectively. Then Y can be written as follows:

$$Y = HX \tag{1}$$

Such that

$$Y_i = \sum_{k=0}^{N-1} H_{ik} X_k \tag{2}$$

If the elements of the matrix transform and the input vector are represented using the 2's complement number representation, then:

$$H_{ik} = -h_{ik,n-1}2^{n-1} + \sum_{l=0}^{n-2} h_{ik,l}2^{l}$$
$$and \quad X_k = -x_{k,n-1}2^{n-1} + \sum_{m=0}^{n-2} x_{k,m}2^{m} \tag{3}$$

where $h_{ik,l}$ and $x_{k,m}$ are the lth bit of H_{ik} and mth bit of X_k, respectively, (which are zero or one) and $h_{ik,n-1}$ and $x_{k,n-1}$ are the sign bits, where n is the word length .

By substituting (3) into (2), the transform coefficient Y_i can be computed as follows:

$$Y_i = \sum_{k=0}^{N-1}\left[-h_{ik,n-1}2^{n-1} + \sum_{l=0}^{n-2} h_{ik,l}2^{l}\right]\left[-x_{k,n-1}2^{n-1} + \sum_{m=0}^{n-2} x_{k,m}2^{m}\right] \tag{4}$$

From equation (4), it can be seen that the computation of the matrix product depends on the type of multiplier used. As mentioned in the introduction a Baugh-Wooley multiplier algorithm [1] has been chosen and using this algorithm equation (4) becomes:

$$Y_i = \sum_{k=0}^{N-1}\left[\sum_{l=0}^{n-2}\sum_{m=0}^{n-2} 2^{l+m}h_{ik,l}x_{k,m} + 2^{2n-2}h_{ik,n-1}x_{k,n-1} + \left(\sum_{l=0}^{n-2} -2^{l}h_{ik,l}x_{k,n-1} + \sum_{m=0}^{n-2} -2^{m}x_{k,m}h_{ik,n-1}\right)2^{n-1}\right] \tag{5}$$

which can be re-arranged as follows, for $(i = 2r)$:

$$Y_{2r} = \sum_{k=0}^{N-1}\left[\sum_{l=0}^{n-2}\sum_{m=0}^{n-2} 2^{l+m}h_{2rk,l}x_{k,m} + 2^{2n-2}h_{2rk,n-1}x_{k,n-1} + \left(\sum_{l=0}^{n-2} 2^{l}\overline{h_{2rk,l}x_{k,n-1}} + \sum_{m=0}^{n-2} 2^{m}\overline{x_{k,m}h_{2rk,n-1}}\right)2^{n-1} + 2^{n} - 2^{2n-1}\right] \tag{6}$$

and for $(i = 2r+1)$:

$$Y_{2r+1} = \sum_{k=0}^{N-1}\left[\sum_{l=0}^{n-2}\sum_{m=0}^{n-2} 2^{l+m}h_{2r+1k,l}x_{k,m} + 2^{2n-2}h_{2r+1k,n-1}x_{k,n-1} + \left(\sum_{l=0}^{n-2} 2^{l}\overline{h_{2r+1k,l}x_{k,n-1}} + \sum_{m=0}^{n-2} 2^{m}\overline{x_{k,m}h_{2r+1k,n-1}}\right)2^{n-1} + 2^{n} - 2^{2n-1}\right] \tag{7}$$

From equation (6) and (7) it can be seen that the multiplication of H_{ik} and X_k expressed in two's complement representation can be written in a form which involves only positive bit products and the matrix-vector product can be computed using the systolic architecture described below.

The two's complement multiplication using the Baugh-Wooley algorithm can be performed using the proposed serial-parallel multiplier as shown in Fig.1 in the case of (n=4). Basically, the (BWM) comprises a logic unit and an adder unit. The logic unit is used to obtain the partial products and adding the extra one in the fifth and the eighth cycle through the OR and the AND gates respectively using the two control signal S1 and S2 as the BWM requires. The adder unit is used for the addition of the partial products and the carry propagation [2].

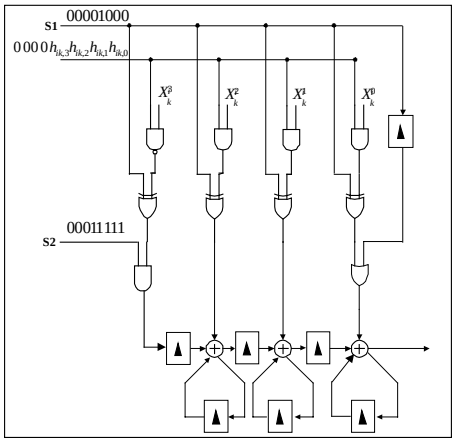

Fig. 1. Baugh-Wooley multiplication (BWM)

2.2. 1-D FHT based Distributed Arithmetic (DA)

Let the input data and the transformed data be represented by the two vectors X and Y of size N, respectively. Then Y can be written as follows:

$$Y = H\,X \tag{8}$$

such that

$$Y_i = \sum_{k=0}^{N-1} H_{ik}X_k \tag{9}$$

where $\{X_k\}$'s are written in the fractional format as shown in equation (10)

$$X_k = -x_{k,n-1} + \sum_{m=1}^{n-1} x_{k,n-1-m}2^{-m} \tag{10}$$

where $x_{k,m}$ is mth bit of x_k (which are zero or one), $x_{k,n-1}$ are the sign bits, where n is the wordlength , respectively.

Substituting (10) in (9):

$$Y_i = \sum_{k=0}^{N-1} H_{ik}\left(-x_{k,n-1} + \sum_{m=1}^{n-1} x_{k,n-1-m}2^{-m}\right) = -\sum_{k=0}^{N-1} H_{ik}x_{k,n-1} + \sum_{m=1}^{n-1}\left(\sum_{k=0}^{N-1} H_{ik}x_{k,n-1-m}\right)2^{-m} \tag{11}$$

Define $$Z_{n-1-m} = \sum_{k=0}^{N-1} H_{ik}x_{k,n-1-m} \quad (m \neq 0)$$

and $$Z_{n-1} = -\sum_{k=0}^{N-1} H_{ik}x_{k,n-1} \qquad (m = 0) \tag{12}$$

The output result is given by:

$$Y_i = \sum_{m=0}^{n-1} Z_{n-1-m}2^{-m} \tag{13}$$

Since the term Z_m depends on the $x_{k,m}$ values it has only 2^N possible values, it is possible to precompute and store these values in a ROM. An input set of N bits $(x_{1,m}, x_{2,m}, ...x_{N,m})$ is used as an address to retrieve the corresponding Z_m values.

A direct implementation of equation (8) would require (in the case of N=8 and n=8) 56 additions and subtractions. However,

with the use of the FHT the number of the addition and subtraction operations is reduced to 24 [5]. To reduce the number of arithmetic operations and to speed up the process, the symmetry in the FHT matrix coefficients and a sparse matrix factorisation are exploited as shown in equations (14),(15) and (16).

$$\begin{aligned} Y &= H\ X \\ &= H^4 H^2 X \\ &= H^4 T \end{aligned} \tag{14}$$

where H^4 and H^2 are uniformly sparse with 4 and 2 non - zero coefficients respectively.

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \\ Y_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \\ Y_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} = \begin{bmatrix} X_1 + X_2 \\ X_1 - X_2 \\ X_3 + X_4 \\ X_3 - X_4 \\ X_5 + X_6 \\ X_5 - X_6 \\ X_7 + X_8 \\ X_7 - X_8 \end{bmatrix} \tag{17}$$

The ROM's contents together with the new input set of *N* bits used as an address to retrieve the corresponding Z_m values are shown in Fig. 2. Where $H^4_{8,ij}$ are the matrix coefficients, characterised by the matrix size=8, 4 non-zero coefficients and the indexes *ij* for the rows and the columns respectively.

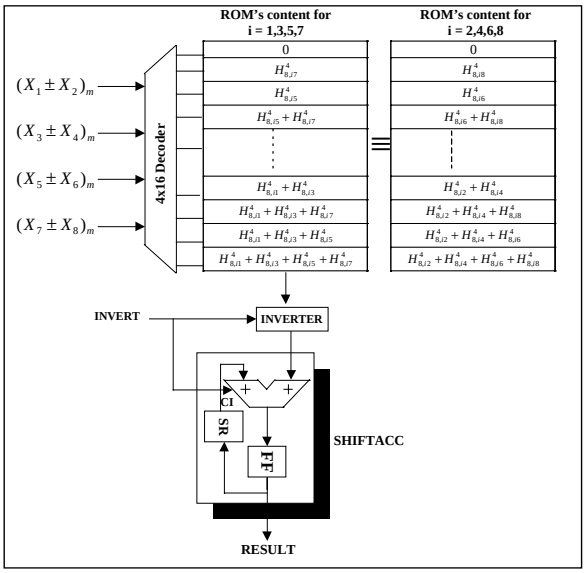

Fig. 2. Distributed Arithmetic ROM and Accumulator (RAC) structure

3. 1-D FHT ARCHITECTURE

In this section the 1-D FHT architectures based on (SA) and (DA) principles are described.

3.1. 1-D FHT based (SA)

Equation (6) and (7) can be mapped into the proposed architecture. Figure 3 shows the architecture obtained for *N*=4 and *n*=4. It consists of eight identical processing elements (PEs). Each PE comprises a serial-parallel Baugh-Wooley Multiplier (BWM) as described in Fig. 1, a Flip Flop (FF) for saving the carry bit and a full adder that adds the result of the partial product and the result generated from the previous PE. The matrix elements H_{ij} are fed from the north in a parallel/serial fashion bit by bit Least Significant Bit First (LSBF) while the vector elements X_j are fed in a parallel fashion and remain fixed in their corresponding PE cell during the entire computation of the operation. Each bit of the final product of the PE is fed to the full adder of the preceding PE so that the corresponding output bit of each PE is added to complete the result bit using LSBF method.

The architecture is symmetric so the odd and the even results can be obtained separately through the two processing stages and by selecting the input samples using the selector which depends on the index *i* values for the input coefficients. In addition, this architecture is regular, modular and can be generalised for any transform length and input word length.

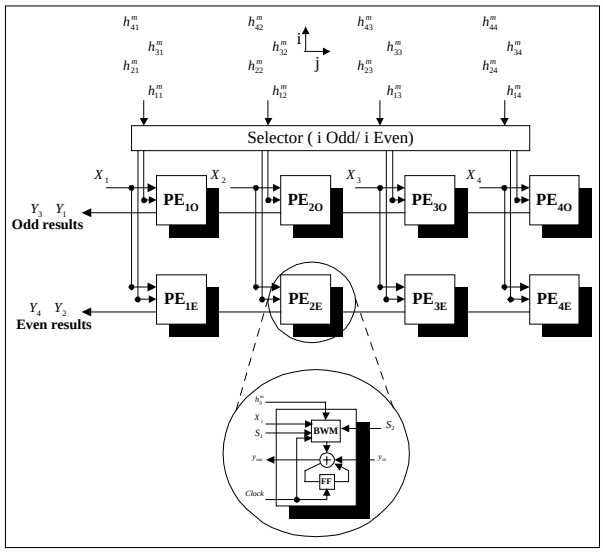

Fig. 3. Proposed systolic architecture for 1-D FHT (*N=4*)

The time complexity and the area of the structure are *n (N+1)T* and *2N*, respectively. (where *T* is the clock cycle fixed by the total gate delay of the BMW multiplier).

Features	Proposed Structure	Structure of [2]	Structure of [4]
Computation time	*(n(N+1))T*	*(2n)T*	*(2nN)T*
Area Complexity	*O(2N)*	*O(N^2)*	*O(N)*

Table 1. Comparison of proposed structure with the existing structures ([2],[4]) for computation of the 1-D FHT

3.2. 1-D FHT based (DA)

The architecture for the 1-D FHT is shown in Fig. 4. The n=8 bit inputs to the circuit are fed in a bit-serial fashion from a converter. The 4 separate RAC blocks calculate the 8 transforms as follows: a butterfly structure of bit-serial adders and subtracters is used to generate the elements of the input matrix in equation (16).

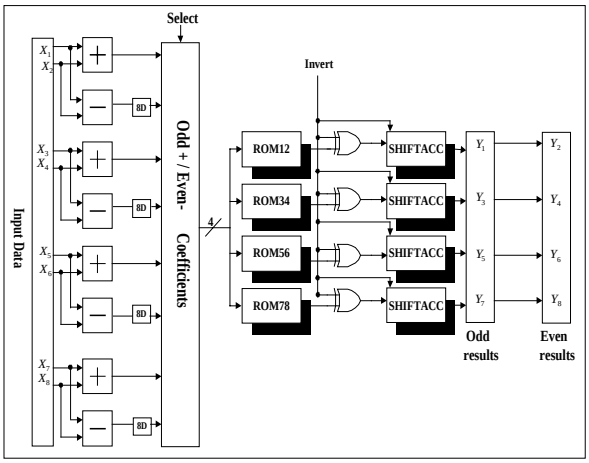

Fig. 4. Proposed architecture for 1-D FHT ($N=8$) based on DA principles

During the first 8 cycles eight bit-parallel outputs for each odd result are produced, and during the second 8 cycles eight bit-parallel outputs for each even results are produced. This is verified when the word length of the bit-serial adders and subtracters results is less or equal (n). Otherwise, an other delay must be added to the 8 delays to complete the operation. This implies a bit-parallel data organization with the maximum value in the ROM represented by 5 bits including the sign extension. Every n clock cycles the signal INVERT is used to compute the two's complement of the ROM's content by inverting the value and inserting a C_{in} (carry in of the adder) of value one while signal SELECT is used to select the odd and even input samples. The selector is basically a multiplexer (8→4), and the computation process runs from $m=0$ to $m = 2\times(n-1)$.

Feature	Proposed Structure	Structure of [3]	Structure of [6]
Computation time	*2n*	*(2N-1) (n+log$_2$N)*	*2N-1*

Table 2. Comparison of proposed structure with the existing structures ([3],[6]) for computation of 1-D FHT

4. FPGA IMPLEMENTATION

The proposed architectures described above have been implemented for ($N=8$, $n=8$) using a Xilinx Virtex XCV1000E FPGA series board with Target Package: fg680. The designs were carried out using the same Relative LoCations (RLOC) attributes to obtain efficient placement.

The designs are modular, regular and can be implemented for larger transform and input data word lengths. Table.3 illustrates the performance obtained for the proposed architectures and the architecture proposed in [4] for the case of N= 8 and n=8. (DA) technique shows significant improvements and better performances when is compared with (SA) technique and [4] concerning the speed and the area consumed by the design. In addition, the proposed systolic architecture provides a high throughput rate when compared with the proposed structure in [4].

Architectures	Slices	I/O	Flip-Flops	4-input LUT	Speed (MHz)
SA	188	134	320	280	31
DA	124	116	169	112	90
[4]	136	132	192	168	45

Table 3. Implementation report for DA technique, SA technique and [4] ($N=8$, $n=8$)

5. CONCLUSION

Due to the importance of the 1-D FHT transform in image and signal processing, two novel architectures have been presented in this paper. The first architecture is based on SA technique while the second one is based on DA principles together with the exploitation of the transform symmetry and sparse matrix factorisation. The effectiveness of the two approaches has been discussed and shown that DA approach provides better performances concerning the speed and the area when is compared with the SA approach.

6. REFRENCES

[1] K. Parhi, "VLSI Digital Signal Processing Systems Design and Implementation." *Wiley*, 1999.

[2] SS. Nayak., PK. Meher, "High throughput VLSI implementation of discrete orthogonal transforms using bit-level vector-matrix multiplier." *IEEE Trans.on Circ.& Syst. II, Analog and Digital Sig. Proc.*, 1999, Vol.46, No.5, pp.655-658.

[3] L.Wen Chang and M.Chang Wu, "A bit level systolic array for Walsh-Hadamard transforms." *Signal Processing* Vol 31, pp 341-347, 1993

[4] A. Amira, A. Bouridane, P. Milligan and P. Sage "A High Throughput FPGA Implementation of A Bit-Level Matrix Product." *Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS), pp 356-364*, October 2000, Lousiana, USA.

[5] A. Amira, A. Bouridane and P. Milligan "A Novel Architecture for Walsh Hadamard Transforms Using Distributed Arithmetic Principles." *Proceedings of the 7th IEEE International Conference on Electronics, Circuits & Systems (ICECS'2K), Vol1, pp182-185,*December 2000, Beirut, Lebanon.

[6] S.Y. Kung, "VLSI Array Processors." Prentice Hall, 1988.

[7] D. Coppersmith and al "Hadamard transforms on multiply/add architecture," *IEEE Trans. Signal Processing, Vol 42, N^o 4, pp. 969-970,* April 1994

[8] URL: www.xilinx.com.

[9] S.Rahardja and B.J. Falkowski "Family of Unified Complex Hadamard Transforms." *IEEE Trans. Circuits and Systems -II: Analog and Digital Signal Processing. Vol.46. N^o.8.* August 1999.

[10] D. Coppersmith and al "Hadamard transforms on multiply/add architecture," *IEEE Trans. Signal Processing, Vol 42, N^o 4,, pp. 969-970,* April 1994